AGP - TP de soutien
Récursivité

Durée approximative : 2 heures

Ce TP a pour but de vous familiariser avec la récursivité. La récursivité est très utilisée en programmation, en particulier dans les langages fonctionnels.

1 PGCD

L'algorithme d'Euclide pour calculer le plus grand commun diviseur de deux entiers naturels a et b (au moins un des deux nombres n'est pas nul) utilise la propriété suivante (*mod* est l'opération modulo) :

$$\text{si } a > b \text{ alors } \begin{cases} \text{si } a \ mod \ b = 0, PGCD(a,b) = b \\ \text{sinon } PGCD(a,b) = PGCD(b, a \ mod \ b). \end{cases}$$

L'algorithme que vous allez implémenter est le suivant :

```
// PGCD(a,b) renvoi le plus grand commun diviseur de deux entiers
ENTIER PRODECURE PGCD(ENTIER a, ENTIER b)
    VARIABLE ENTIER res
DEBUT
    SI (a>b) ALORS
        SI (a mod b == 0) ALORS
            res ← b
        SINON
            res ← PGCD(b, a mod b)
        FINSI
    SINON
        SI (b mod a == 0) ALORS
            res ← a
        SINON
            res ← PGCD(a, b mod a)
        FINSI
    FINSI
    RETOUR(res)
FIN
```

1.1 Implémentation en C

Implémenter une fonction `pgcd` en C, en utilisant l'algorithme proposé. On pourra utiliser l'opérateur modulo (`%`). On mettra en place une fonction `main` qui appelle cette fonction et on écrira le Makefile permettant de compiler l'ensemble.

1.2 Visualisation des appels récursifs

Afin de visualiser l'emboîtement des appels récursifs, modifier cette fonction afin d'écrire :

— En début d'exécution de la fonction, un message indiquant le début du calcul et les valeurs des paramètres a et b ;

— En fin d'exécution de la fonction, un message indiquant la fin du calcul et la valeur retournée par la fonction.

Essayez de mettre une identation différente pour chaque appel récursif pour bien visualiser ce que chaque appel exécute. Exemple d'affichage qui pourrait être produit par votre fonction :

```
 Appel de pgcd(779,665)
     Appel de pgcd(665,114)
         Appel de pgcd(114,95)
             Appel de pgcd(95,19)
             retour de pgcd(95,19)=19
         retour de pgcd(114,95)=19
     retour de pgcd(665,114)=19
 retour de pgcd(779,665)=19
Le pgcd de 779 et 665 est... 19
```

On remarque que le résultat est calculé par le dernier appel récursif et transmis tel quel à la fonction appelante, jusqu'à l'appel initial de la fonction `main` qui affiche le résultat. Il s'agit d'une récursion *terminale* : aucun traitement n'est fait sur le résultat de l'appel récursif. Les récursions terminales sont très efficacement compilées par les compilateurs, savez vous pourquoi ?

2 Carré d'un nombre

Utiliser l'identité $n^2 = (n-1)^2 + 2n - 1$ pour écrire un programme récursif qui calcule le carré d'un entier positif ou nul n. Afficher, comme précédemment, l'argument lors de l'appel de la fonction et le résultat avant de sortir de la fonction. On remarque que le résultat est différent à chaque appel, ce n'est pas une récursion terminale.

3 Coefficient binomial

Le coefficient binomial des entiers naturel n et k noté $\begin{pmatrix} n \\ k \end{pmatrix}$ (anciennement C_n^k : combinaison de k parmi n) peut être calculé récursivement avec les informations suivantes :

$$\begin{pmatrix} n \\ k \end{pmatrix} \text{ vaut 1 si } k = n \text{ ou } k = 0$$
$$\begin{pmatrix} n \\ k \end{pmatrix} + \begin{pmatrix} n \\ k+1 \end{pmatrix} = \begin{pmatrix} n+1 \\ k+1 \end{pmatrix} \text{ si } 0<\text{k}<\text{n}$$

Écrire un programme qui, par une double récursion calcule $\begin{pmatrix} n \\ k \end{pmatrix}$, l'exécutable prendra en paramètres les deux entiers n et k sue la ligne de commande.